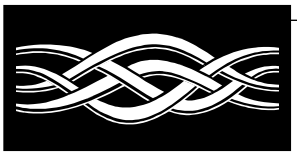
Microsoft®

Microsoft® Windows NT® Server

System Operating System



White Paper

Active Directory Technical Summary.

© 1997 Microsoft Corporation. All rights reserved.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

X and Y are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Other tademarks or tradenames mentioned herein may be the trademarks of their respctive owners.

Microsoft Corporation • One Microsoft Way • Redmond, WA 98052-6399 • USA

XX96 Part no. 098-XXXXX

Microsoft® Windows NT® Server

Abstract

This document provides a technical overview of the Active Directory, including important concepts and features, an introduction to the Active Directory architecture, information on migrating from Windows NT 4.0 to Windows NT 5.0, and a frequently asked questions (FAQ) section. The Active Directory is the foundation for the Windows NT Distributed System, which is the platform on which Microsoft’s BackOffice family of integrated products is built.

CONTENTS

Introduction....4
What is a Directory Service? 4
Why Have a Directory Service? 4
What is the Active Directory? 5

Important Concepts....6
Scope 6
Namespace 6
Object 6
Container 7
Tree 7
Name 8
  Distinguished Name 8
  Relative Distinguished Name 9
Naming Contexts and Partitions 9
Domains 9
Domain Trees 9
  Viewing Trust Relationships 9
  Viewing the Namespace 10
Forests 11
Sites 11

Architecture....12
Directory System Agent 12
Data Model 12
Schema 12
Security Model 12
Administration Model 12

Active Directory Features....14
DNS Integration 14
  Location Service 14
  Dynamic DNS 14
Object Naming 15
  Uniqueness of Names 15
Access 15
  Protocols 15
  Application Programming Interfaces 16
Virtual Containers 17
Global Catalog 17
Security 17
  Delegation 18
  Inheritance 18
Replication 19

Update Propagation 19
Collision Detection – Version Numbers 19
Propagation Dampening 20
Trees and Forests 20
Transitive Bidirectional Trust 20
Namespace 21
When To Form a Domain Tree 22
How To Form a Domain Tree or Forest 22
Sites 22
Sites and Replication 23
Planning Sites 23
Schema 23
When To Extend the Schema 23
Adding Attributes 23
Adding New Objects 24
How to Extend the Schema 24
Object Identifiers (OIDs) 24
Publishing 25
When To Publish 25
How To Publish 26
Groups 26

Migration....27
Supported Upgrade Paths 27
Domain Controllers 27
Member Servers 28
Clients 28
User Accounts 28
Machine Accounts 28
Global Groups 28

Frequently Asked Questions....29
How does a workstation discover its site? 29
How does a workstation find a directory server? 29
How do I log in? 29
What happens to Access Control Lists on domain resources after migration? 29
What happens to ACL’s when I delete a domain? 30
What happens to local groups? 30
Part 1: Local Groups on Member Servers 30
Part 2: Local Groups on PDCs and BDCs 30
When is the Global Catalog searched? 31
Do I have to use Microsoft’s DNS Server? 31
What are the advantages of using Microsoft’s DNS Server? 31
What happens to DHCP? 31

What happens to WINS? 31

For more information.....33

This document provides a technical introduction to the Active Directory, the new directory service provided by Windows NT Server version 5.0. This document includes detailed explanations of important Active Directory concepts, architectural elements, and features. The following section, Important Concepts, describes the terms you'll need to understand before tackling the Active Directory itself. The next two sections, Active Directory Features and Architecture, go into more detail on what the Active Directory accomplishes, what features it brings to Windows NT, and how it's implemented. The next to last section, Migration, covers migrating domain models and directory structures from Windows NT 4.0 to Windows NT 5.0. The last section, Frequently Asked Questions, answers a set of practical questions about the Active Directory and how it works.

What is a Directory Service?

A *directory* is an information source used to store information about interesting objects. A telephone directory stores information about telephone subscribers. In a file system, the directory stores information about files.

In a distributed computing system or a public computer network like the Internet, there are many interesting objects, such as printers, fax servers, applications, databases, and other users. Users want to find and use these objects. Administrators want to manage how these objects are used.

In this document the terms *directory* and *directory service* refer to the directories found in public and private networks. A *directory service* differs from a *directory* in that it is both the directory information source *and* the services making the information available and usable to the users.

Why Have a Directory Service?

A directory service is one of the most important components of an extended computer system. Users and administrators frequently do not know the exact name of the objects they are interested in. They may know one of more *attributes* of the objects and can *query* the directory to get a list of objects that match the attributes. For example, "find all duplex printers in Building 26." A directory service

allows a user to find any object given one of its attributes.

A directory service can:

- Enforce *security* defined by administrators to keep information safe from intruders.
- *Distribute* a directory across many computers in a network.
- *Replicate* a directory to make it available to more users and resistant to failure.
- *Partition* a directory into multiple stores to allow the storage of a very large numbers of objects.

A directory service is both a management tool and an end user tool. As the number of objects in a network grows, the directory service becomes essential. The directory service is the hub around which a large distributed system turns.

What is the Active Directory?

The Active Directory is the directory service included with Windows NT Server version 5.0. It extends the features of previous Windows-based directory services and adds entirely new features. The Active Directory is *secure, distributed*, *partitioned*, and *replicated*. It is designed to work well in any size installation, from a single server with a few hundred objects to thousands of servers and millions of objects. The Active Directory adds many new features that make it easy to navigate and manage large amounts of information, generating savings for both administrators and end users.

Some concepts and terms that are used to describe the Active Directory are new and some aren’t. Unfortunately, some of the terms that have been around for a while are used to mean more than one particular thing. Before going on, it is important that you understand how the following concepts and terms are meant in the context of the Active Directory.

Scope

The scope of the Active Directory is large. It can include every single object (printer, file, or user), every server, and every domain in a single wide area network. It can also include several wide area networks combined. Some of the following terms apply to more than a single network, so it is important to keep in mind that the Active Directory can scale from a single computer, to a single computer network, to many computer networks combined.

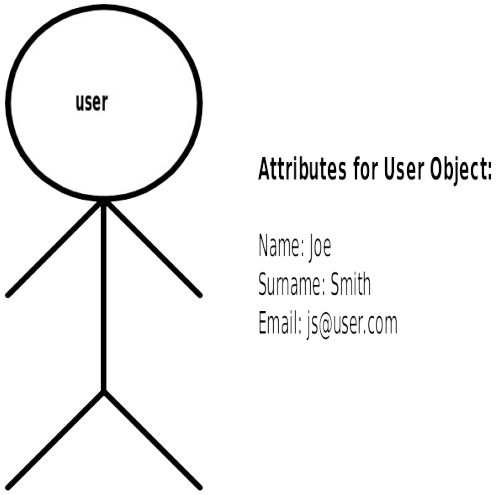
Namespace

The Active Directory is primarily a *namespace*, as is any directory service. A telephone directory is a namespace. A namespace is any bounded area in which a given name can be resolved. Name resolution is the process of translating a name into some object or information that the name represents. A telephone book forms a namespace in which the names of telephone subscribers can be resolved to telephone numbers. The Windows NT file system forms a namespace in which the name of a file can be resolved to the file itself.

The Active Directory forms a namespace in which the name of an object in the directory can be resolved to the object itself.

Object

An *object* is a distinct, named set of *attributes* that represents something concrete, such as a user, a printer, or an application. The attributes hold data describing the thing that is identified by the directory object. Attributes of a user might include the user's given name, surname, and e-mail address.



<adfig1.vsd, stick figure, attributes name, surname, email address>

Figure 1. A user object and its attributes.

Container

A *container* is like an object in that it has attributes and is part of the Active Directory namespace. However, unlike an object, it does not represent something concrete. It is the container of a group of objects and other containers.

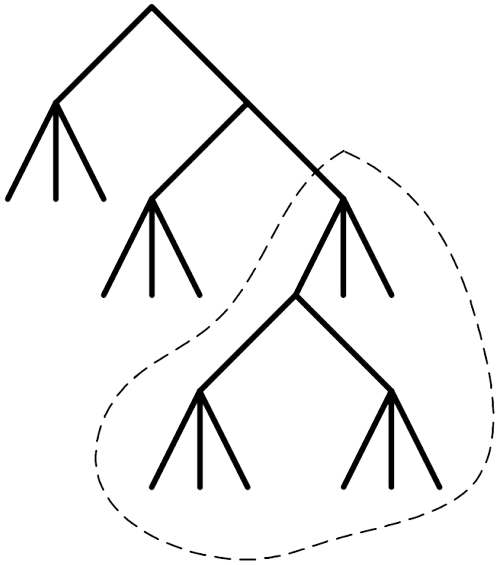
Distinguished Name

Tree

Tree is used throughout this document to describe a hierarchy of objects and containers. Endpoints on the tree are usually objects. Nodes or branches are containers.

A tree shows how objects are connected or the path from one object to another. A simple directory is a container. A computer network or domain is also a container.

A contiguous subtree is any unbroken path in the tree including all



the members of any container in that path.

<adfig2.vsd, a picture of a directory tree with a contiguous portion outlined.>

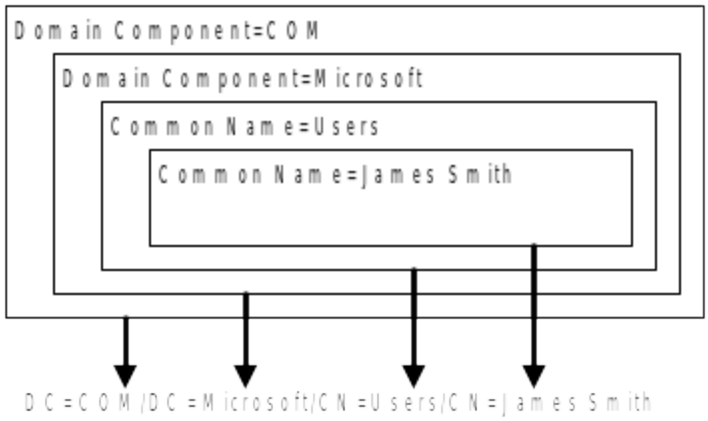
Figure 2. A contiguous subtree of a file directory.

Name

A name is used to identify every object in the Active Directory. There are two different kinds of names.

Every object in the Active Directory has a *distinguished name* (DN). The distinguished name identifies the domain that holds the object as well as the complete path through the container hierarchy by which the object is reached. A typical DN might be /O=Internet/DC=COM/DC=Microsoft/CN=Users/CN=James Smith. This DN identifies the “James Smith” user object in the Microsoft.Com domain.

Relative Distinguished Name



<adfig3.vsd, show each level of container as an area and expand letters, Organization=Internet, etc.>

Figure 3. A graphical representation of a distinguished name.

The *Relative Distinguished Name* (RDN) of an object is the part of the name that is an attribute of the object itself. In the preceding example, the RDN of the “James Smith”` user object is “CN=James Smith”. The RDN of the parent object is “CN=Users”.

Naming Contexts and Partitions

The Active Directory is made up of one or more *naming contexts* or *partitions.* A naming context is any contiguous subtree of the directory. Naming contexts are the unit of replication.

In the Active Directory a single server always holds at least three naming contexts:

- The schema.
- The configuration (replication topology and related metadata).
- One or more user naming contexts (subtrees containing the actual objects in the directory).

Domains

A *domain* is a single security boundary of a Windows NT computer network. (For more information on Windows NT domains, see your Windows NT documentation.) The Active Directory is made up of one or more domains. On a standalone workstation, the domain is the computer itself. A domain can span more than one physical location. Every domain has its own security policies and security relationships with other domains. When multiple domains are connected by trust relationships and share a common schema, configuration, and global

catalog, you have a *domain tree*. Multiple domain trees can be connected together into a *Forest.*

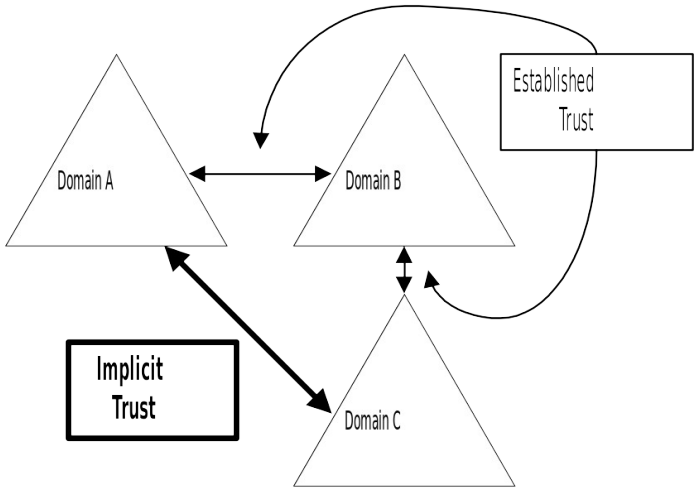
Domain Trees

A domain tree ("Tree") is comprised of several domains that share a common schema and configuration, forming a contiguous namespace. Domains in a tree are also linked together by trust relationships. The Active Directory is a set of one or more ~~domain t~~Trees.

~~Domain t~~Trees can be viewed two ways. One view is the trust relationships between domains. The other view is the namespace of the domain tree.

You can draw a picture of a domain tree based on the individual domains and how they trust each other.

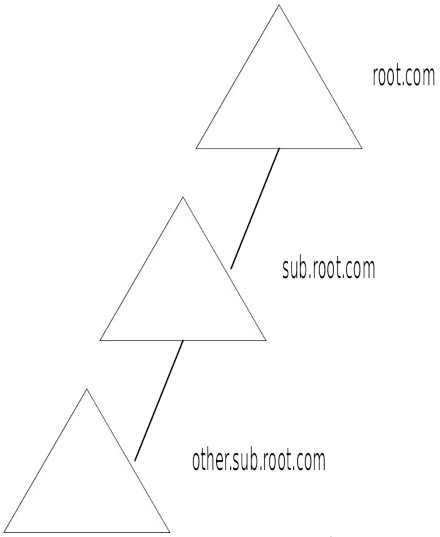
Windows NT establishes trust relationships between domains based on the Kerberos security protocol. Kerberos trust is transitive and hierarchical—if domain A trusts domain B and domain B trusts domain C, domain A trusts domain C as well.



<adfig4.vsd, trust tree between domains>

Figure 4. A domain tree viewed in terms of its trust relationships.

You can also draw a picture of a domain tree based on the namespace. You can determine an object's distinguished name by following the path up the domain tree's namespace. This view is useful for grouping objects together into a logical hierarchy. Its chief advantage of a contiguous namespace is that a deep search from the root of the namespace will search the entire hierarchy.~~all the~~



~~names in the namespace, even if individual trees are not contiguous.~~

<adfig5.vsd, show domain tree in terms of names, also show how physical discontiguous names ~~can still form namespace~~form a forest.>

Figure 5. Viewing a domain tree as a namespace.~~ Though these domains may be physically discontiguous, they share a single, common namespace~~.

~~The Active Directory does not limit forming a contiguous namespace from discontiguous domains and directories. However, it may be advantageous to form contiguous trees that match the namespace such that replication follows the same logic that the namespace presents.~~

Forests

A *Forest* is a set of one or more Trees that *do not* form a contiguous namespace. All Trees in a Forest share a common schema,

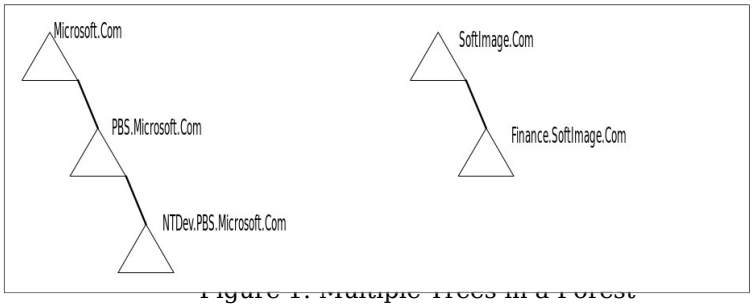


Figure 4. Multiple Trees in a Forest

configuration, and Global Catalog. All trees in a given Forest trust each other via transitive hierarchical Kerberos trust relationships. Unlike Trees, a forest does not need a distinct name. A forest exists as a set of Cross-reference objects and Kerberos trust relationships known to the member trees. Trees in a Forest form a hierarchy for the purposes of Kerberos trust; the Tree name of at the root of the trust tree can be used to refer to a given Forest.

Sites

A *site* is a location in a network holding Active Directory servers. A site is defined as one or more well connected TCP/IP subnets. "Well connected" means that network connectivity is highly reliable and fast (e.g. LAN speeds, 10 MM bits-per-second or greater). Defining a site as a set of subnets allows administrators to quickly and easily configure the Active Directory access and replication topology to take advantage of the physical network. When users log in, Active Directory clients find Active Directory servers in the same site as the user. Since machines in the same Site are close to each other in network terms, communication among machines in the is reliable, fast, and efficient.~~This~~ Determining the local Site at login time is easy because the user's workstation already knows what TCP/IP subnet it is on, and subnets translate directly to Active Directory sites.

Directory System Agent

This short section introduces some of the primary architectural components of the Active Directory.

Data Model

The Active Directory data model is derived from the X.500 data model. The directory holds objects that represent things of various sorts, described by attributes. The universe of objects that can be stored in the directory is defined in the schema. For each object class, the schema defines what attributes an instance of the class must have, what additional attributes it may have, and what object class can be a parent of the current object class.

Schema

The Active Directory schema is implemented as a set of object class instances stored in the directory. This is very different than many directories which have a schema but store it as a text file read at

startup. Storing the schema in the directory has many advantages. For example, user applications can read it to discover what objects and properties are available.

The Active Directory schema can be updated dynamically. That is, an application can extend the schema with new attributes and classes and use the extensions immediately. Schema updates are accomplished by creating or modifying the schema objects stored in the directory. Like every object in the Active Directory, schema objects are protected by ACLs, so only authorized users may alter the schema.

Security Model

The directory is part of the Windows NT Trusted Computing Base and is a full participant in the Windows NT security infrastructure. Access Control Lists (ACLs) protect all objects in the Active Directory. Any attempt to access an object or attribute in the Active Directory is validated against the ACL by the Windows NT access validation routines.

Administration Model

Authorized users perform administration in the Active Directory. A user is authorized by a higher authority to perform a specified set of actions on a specified set of objects and object classes in some identified subtree of the directory. This is called *delegated administration.* ~~This~~ Delegated administration allows very fine-grained control over who can do what and enables delegation of authority without granting elevated privileges.

The Directory System Agent (DSA) is the process that manages the directory's physical storage. Clients connect to the DSA using one of the supported interfaces to search for, read, and write the directory objects and their attributes. The DSA provides isolation of the clients from the physical storage format of the directory data.

This section describes some of the major features and components of the Active Directory.

DNS Integration

The Active Directory is tightly integrated the Domain Name System (DNS). DNS is the distributed namespace used on the Internet to resolve computer and service names to TCP/IP addresses. Most

Location Service

enterprises with intranets use DNS as the name resolution service. The Active Directory uses DNS as the *location service.* Windows NT 5.0 domain names are DNS domain names. For example "Microsoft.Com" is a valid DNS domain name and could also be the name of a Windows NT 5.0 Domain. Tight DNS integration means the Active Directory fits naturally into Internet and intranet environments. Clients find directory servers quickly and easily. An enterprise can connect Active Directory servers directly to the Internet to facilitate secure communications and electronic commerce with customers and partners.

Active Directory servers publish their addresses such that clients can find them knowing only the domain name. Active Directory servers are published via Service Resource Records (SRV RRs) in DNS[1]. The SRV RR is a DNS record used to map the name of a service to the address of a server offering that service. The name of a SRV RR is in this form:

```
<service>.<protocol>.<domain>
```

Active Directory servers offer the LDAP service over the TCP protocol so that published names are "ldap.tcp.<domain>". Thus, the SRV RR for "Microsoft.Com" is "ldap.tcp.microsoft.com". Additional information on the SRV RR indicates the priority and weight for the server, allowing clients to choose the best server for their needs.

When an Active Directory server is installed, it publishes itself via *Dynamic DNS*. Since TCP/IP addresses are subject to change over time, servers periodically check their registrations to make sure they are correct, updating them if necessary.

Dynamic DNS

Dynamic DNS is a recent addition to the DNS standard[2]. Dynamic DNS defines a protocol for updating a DNS server with new or changed values dynamically. Prior to Dynamic DNS administrators needed to manually configure the records stored by DNS servers.

Object Naming

An object has exactly one name, the Distinguished Name (DN). The DN uniquely identifies the object and contains sufficient information for a client to retrieve the object from the directory. The DN of an

[1] ~~RFC????~~

[2] RFC 2136.R~~FC????. This RFC is proposed for standard status and is expected to receive formal recognition as a standard in mid-1997.~~

object may be quite long and difficult to remember. Moreover, the DN of an object may change. Since the DN of an object is composed of the RDN of the object and its ancestors, a rename of the object itself or any ancestor will change the DN.

Since DNs are complex to remember and subject to change, it is useful to have other means for retrieving objects. The Active Directory supports querying by attributes, so an object can be found even if the exact DN is unknown or has changed. To simplify the process of finding objects by query, the Active Directory schema defines two useful properties[3] ~~for all objects~~:

Object GUID – A 128-bit number, guaranteed to be unique. All~~O~~ objects have a GUID assigned when they are created. The GUID is never changed, even if the object is moved or renamed. Applications can store the GUID of an object and be assured of retrieving that object no matter what the current DN is.

User Principal Name

Security Principals (users and groups) have a "friendly" name, the *User Principal Name,* shorter than the DN and easier to remember. The User Principal Name is composed of a "shorthand" name for the user and the DNS name of the domain tree where the user object resides. For example, user James Smith in the Microsoft.Com tree might have a UPN of "**JamesS@Microsoft.Com".** The UPN is the preferred login name for Windows NT 5.0 users. UPN is independent of the user's Distinguished Name, so a user's object can be moved or renamed without affecting their login name.

~~**Friendly Name** – An object can be given a friendly name, typically shorter than the DN and easier to remember. The friendly name on a user object can be used to log in, if it is unique.~~

Distinguished names are guaranteed to be unique. The Active Directory does not permit two objects with the same RDN under the same parent. DNs are composed of RDNs, and are therefore unique. GUIDs are unique by definition; an algorithm that ensures uniqueness generates GUIDs. Uniqueness is not enforced for any other attributes.

Access

Access to the Active Directory is via *wire protocols*. Wire protocols define the formats of messages and interactions of client and server. Various application programming interfaces (APIs) give developers

[3] The schema defines many useful properties, these are particularly useful in searching.

Protocols Application Programming Interfaces

access to these protocols.

LDAP—The Active Directory core protocol is the Lightweight Directory Access Protocol (LDAP). LDAP version 2 and version 3[4] are supported.

MAPI-RPC—The Active Directory supports the remote procedure call (RPC) interfaces supporting the MAPI interfaces.

X.500—The Active Directory information model is derived from the X.500 information model. X.500 defines several wire protocols:

DAP – Directory Access Protocol
DSP – Directory System Protocol
DISP – Directory Information Shadowing Protocol
DOP – Directory Operational Binding Management Protocol

The Active Directory **does not implement** these protocols, for several reasons:

- There is very little interest in these protocols, and there are few implementations.
- These protocols are dependent on OSI networking. OSI is an alternative to TCP/IP, which is not widely implemented. Transporting OSI across a TCP/IP network is less efficient than using TCP/IP directly.
- LDAP provides the most important functions offered by DAP and DSP, and is designed to work over TCP/IP without the overhead of "enveloping" OSI in TCP/IP.
- There is sufficient ambiguity in the 1993 and 1997 DISP and DOP specifications that conformant implementations are not guaranteed to interoperate, dramatically reducing the value of these protocols.

~~If there is sufficient demand, Microsoft will implement subsets of these X.500 protocols.~~

ADSI—Active Directory Service Interfaces (ADSI) provide a simple, powerful object-oriented interface to the Active Directory[5]. Developers can use many different programming languages including Java, Visual Basic, C, C++, and others. ADSI is fully scriptable for ease of use by system administrators. ADSI hides the details of LDAP communications from users.

---

[4] LDAP v3 (RFC 2251) is a~~n Internet Draft~~ Proposed Standard as of this writing (8/12/985~~/9/97~~).

[5] ADSI provides common access to multiple directories, including all LDAP directories, Windows NT 4.0, NetWare 3.x, and NetWare 4.x.

LDAP API—The LDAP ‘C’ API, defined in RFC 1823 is a lower-level interface available to ‘C’ programmers.

MAPI—The Active Directory supports MAPI for backward compatibility. New applications should use ADSI or the LDAP ‘C’ API.

Virtual Containers

The Active Directory allows other directories to be exposed via *Virtual Containers.* A Virtual Container allows any LDAP-compliant directory to be accessed transparently via the Active Directory. The Virtual Container is implemented via "knowledge information" stored in the Active Directory. The knowledge information describes where in the Active Directory the foreign directory should appear, and contains the DNS name of a server holding a copy of the foreign directory and the distinguished name (DN) at which to begin search operations in the foreign DS.

Global Catalog

The Active Directory can consist of many partitions or naming contexts. The Distinguished Name (DN) of an object includes enough information to locate a replica of the partition that holds the object. Many times, however, the user or application does not know the DN of the target object or which partition might contain the object. The *Global Catalog* (GC) enables users and applications to find objects in an Active Directory domain tree given one or more attributes of the target object.

The Global Catalog contains a partial replica of every user naming context in the directory. It contains the schema and configuration naming contexts as well. This means the GC holds a replica of every object in the Active Directory but with only a small number of their attributes. The attributes in the GC are those most frequently used in search operations (such as a user’s first and last names, login names, etc.) and those required to locate a full replica of the object. The GC allows users to quickly find objects of interest without knowing what domain holds them and without requiring a contiguous extended namespace in the enterprise.

The Global Catalog is built automatically by the Active Directory replication system. The replication topology for the Global Catalog is generated automatically. The properties replicated into the Global Catalog include a base set defined by Microsoft. Administrators can specify additional properties to meet the needs of their installation.

Security

This is only an overview of security in the Active Directory. The

Delegation

Windows NT 5.0 security model is described in more detail in a separate document, http://premium.microsoft.com/msdn/library/conf/pdc97/distributed_security_services.htm~~<<TBS – security whitepaper title>>.~~

Object Protection

All objects in the Active Directory are protected by an Access Control List (ACL). ACLs determine who can see the object and what actions each user can perform on the object. The existence of an object is never revealed to a user who is not allowed to see it.

An ACL is a list of Access Control Entries (ACEs) stored with the object it protects. In Windows NT an ACL is stored as a binary value, called a Security Descriptor. Each ACE contains a Security Identifier (SID), which identifies the *principal* (user or group) to whom the ACE applies, and information on what type of access the ACE grants or denies.

ACLs on directory objects contain ACEs that apply to the object as a whole and ACEs that apply to the individual attributes of the object. This allows an administrator to control not just which users can see an object, but what properties those users can see. For example, all users might be granted read access to the e-mail and telephone number attributes for all other users, but security properties of users might be denied to all but members of a special security administrators group. Individual users might be granted write access to personal attributes such as the telephone and mailing addresses on their own user objects.

Delegation is one of the most important security features of the Active Directory. Delegation allows a higher administrative authority to grant specific administrative rights for containers and subtrees to individuals and groups. This eliminates the need for “Domain Administrators” with sweeping authority over large segments of the user population.

ACEs can grant specific administrative rights on the objects in a container to a user or group. Rights are granted for specific operations on specific object classes via ACEs in the container’s ACL. For example, to allow user “James Smith” to be an administrator of the “Corporate Accounting” organizational unit, you would add ACEs to the ACL on “Corporate Accounting” as follows[6]:

```
“James Smith”;Grant ;Create, Modify, Delete;Object-Class
```

[6] These “ACEs” are for illustration only, the actual syntax of an ACE will be different than that~~t~~ given in the example.

```
Inheritance    Update Propagation
User
“James Smith”;Grant ;Create, Modify, Delete;Object-Class
Group
“James Smith”;Grant ;Write;Object-Class User; Attribute
Password
```

Now James Smith can create new users and groups in Corporate Accounting and set the passwords on existing users, but he cannot create any other object classes and he cannot affect users in any other containers (unless, of course, he is granted that access by ACEs on the other containers.

Inheritance allows a given ACE to be propagated from the container where it was applied to all children of the container. Inheritance can be combined with delegation to grant administrative rights to a whole subtree of the directory in a single operation.

Replication

The Active Directory provides *multi-master replication*. Multi-master replication means that all replicas of a given partition are writeable. This allows updates to be applied to any replica of a given partition. The Active Directory replication system propagates the changes from a given replica to all other replicas. Replication is automatic and transparent.

Some directory services use timestamps to detect and propagate changes. In these systems, it is very important to keep the clocks on all directory servers synchronized. Time synchronization in a network is very difficult. Even with very good network time synchronization it is possible for the time at a given directory server to be incorrectly set. This can lead to losing changes.

Windows NT 5.0 provides distributed time synchronization, but the Active Directory replication system does not depend on time for update propagation. Instead the Active Directory replication system uses *Update Sequence Numbers* (USN’s). A USN is a 64-bit number maintained by each Active Directory server. When the server writes any property to the Active Directory, the USN is advanced and stored with the property written. This operation is performed atomically—that is, the incrementing and storage of the USN and the write of the property succeed or fail as a single unit of work.

Each Active Directory server also maintains a table of USN’s received from replication partners. The highest USN received from each partner is stored in this table. When the Directory Server is notified by a given partner that replication is required, that server requests

all changes with USN's greater than the last value received. This is a very simple approach, and does not depend on the accuracy of timestamps.

Since the USN stored in the table is updated atomically for each update received, recovery after a failure is therefore also simple. To restart replication a given server simply asks its partners for the changes with USN's greater than the last valid entry in the table. Since the table is updated atomically as the changes are applied, an interrupted replication cycle will always pick up exactly where it left off, with no loss or duplication of updates.

In a multi-master replication system like the Active Directory, it is possible for the same property to be updated at 2 or more different replicas. When a property changes in a second (or third, etc.) replica before a change from the first replica has been fully propagated a *replication collision* occurs. Collisions are detected through the use of *Property Version Numbers.* Unlike USN's, which are server-specific values, the property version number is specific to the property on an Active Directory object. When a property is first written to a Active Directory object, the version number is initialized. *Originating writes* advance the version number. An originating write is a write to a property at the system initiating the change. Property writes caused by replication are not originating writes and do not advance the version number. For example, when a user updates their password, an originating write occurs and the password version number is advanced. Replication writes of the changed password at other servers do not advance the version number.

A collision is detected when a change is received via replication in which the property version number received is equal to the locally stored property version number, and the received and stored values are different. When this occurs the receiving system will apply the update if it has a later timestamp—This is the only situation that time is used in replication. When the received version number is lower than the locally stored version number, the update is presumed stale and discarded. When the received version number is higher than the locally stored version number, the update is accepted.

The Active Directory replication system allows loops in the replication topology. This allows the administrator to configure a replication topology with multiple paths among the servers for performance and availability. The Active Directory replication system performs *propagation dampening* to prevent changes from propagating endlessly and to eliminate redundant transmission of changes to replicas that are already up to date.

The Active Directory replication system employs *up-to-date vectors* to dampen propagation. The up-to-date vector is a list of server-USN pairs held by each server. The up-to-date vector at each server indicates the highest USN of originating writes received from the server in the server-USN pair. An up-to-date vector for a server in a given site lists all the other servers in that site. When a replication cycle begins, the requesting server sends its up-to-date vector to the sending server. The sending server uses the up-to-date vector to filter changes sent to the requesting server. If the high USN for a given originating writer is greater than or equal to the originating write USN for a particular update, the sending server does not need to send the change; the requesting server is already “up-to-date” with respect to the originating writer.

Trees and Forests

A Windows NT 5.0 domain tree is a hierarchy of Windows NT domains, each consisting of a partition of the Active Directory. The shape of the tree and the relationship of the tree members to each other is determined by the ~~trust relationships between domains (or partitions). There may also be a contiguous namespace that is different in shape from the trust relationships between domains~~DNS names of the domains. The domains in a tree must form a contibuous namespace, such that a.myco.com is a child of myco.com, b.a.myco.com is a child of a.myco.com, and so on..

Transitive Bidirectional~~Kerberos~~ Trust

When a domain is joined to a Windows NT 5.0 domain tree a transitive bidirectional~~Kerberos~~ trust relationship is automatically established between the joined-from domain and its parent in the tree. ~~Kerberos trust~~Because the is transitive and bidirectional,~~ so~~ no additional trust relationships are required among tree members. The trust hierarchy is stored ~~in *cross-reference* objects in the Directory. Cross-reference objects are stored~~ as part of the directory metadata in the Configuration~~Partitions~~ container.

~~The Partitions container is a child of the Configuration container that contains Cross-reference objects. There is a Cross-reference object for each naming context in the enterprise. (Recall that each domain is a naming context.)~~

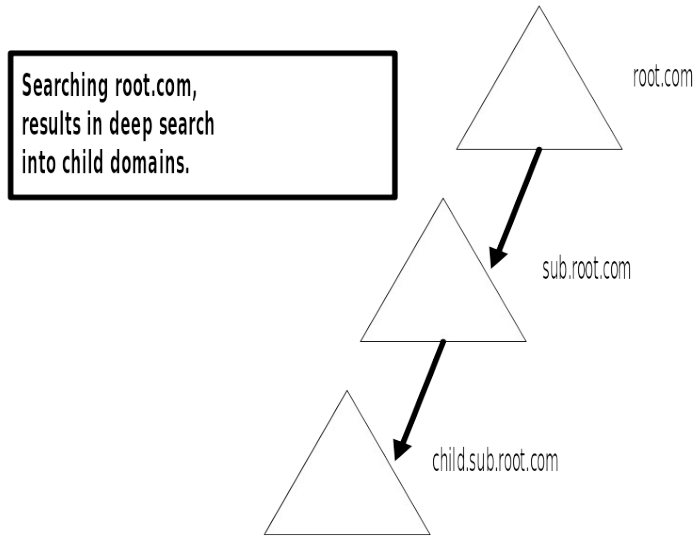
~~There may be Cross-reference objects for virtual containers and for referrals to outside of the enterprise. Cross-reference objects for naming contexts in the enterprise are identified with a flag. Cross-reference objects have an NC-Name attribute that identifies the X.500-style name of the associated domain and a DNS-Root attribute that identifies the DNS name of the associated domain. Trusted domain objects in each domain contain more information about the~~

Namespace When To Form a Domain Tree

~~trust relationship including passwords, direction, and type.~~

Th~~es~~e required objects are created in the directory when a domain is joined to the tree.

The domains in a Windows NT 5.0 domain tree mustay ~~optionally~~ form a contiguous namespace. By default, the immediate children of each domain are contiguous and already part of their namespace. This means that the Distinguished Name of every object in child domains has the name of the parent domain as a prefix. Disjoint trees can be joined into a Forest.



<adfig6.vsd, diagram of this example For example, a parent domain **O=Internet/DC=COM/DC=Microsoft** and a child domain **O=Internet/DC=COM/DC=Microsoft/DC=PBS**, form a contiguous naming tree, because the root object in the parent domain, **O=Internet/DC=COM/DC=Microsoft**, is the immediate parent of the root object in the child, **O=Internet/DC=COM/DC=Microsoft/DC=PBS.** Because the parent and child form a naming tree, a deep search initiated in the parent will automatically search the child as well.>

Figure 6. The parent domain and child domain form a contiguous namespace because the name of the child domain is an immediate subordinate of the name of the parent domain.~~of their inherent trust relationship~~.

The Windows NT 5.0 domain tree *is* the enterprise-wide Active Directory. All Windows NT 5.0 domains in a given enterprise should belong to the enterprise domain tree. Enterprises that need to support disjoint DNS names for their domains will form a Forest.

Windows NT 5.0 domains are joined to a domain tree during the installation process. During the installation of a new Windows NT 5.0 server (or upgrade from an earlier version of Windows NT), the user is given the option of creating the first tree in a new forest, creating a new tree in an existing forest, creating a new replica of an existing domain, or installing a child domain. To join a domain tree, select the Install a Child Domain option and identify the parent domain of which the new domain will be a child. A future update to Windows NT 5.0 will add the capability of joining two (or more) existing trees together into a single, larger tree.

Domains within an existing tree can be freely moved to change the overall tree shape. Planning a good tree is very important, but what is good is highly subjective and based on the specific needs of your organization. The ability to reshape the tree as needed reduces the importance of knowing the right design in advance.

Sites

A *site* is an area of the network where connectivity among machines is assumed to be very good. Windows NT 5.0 defines a site as one or more IP subnets. This is based on the assumption that computers with the same subnet address are connected to the same network segment, typically a LAN or other high-bandwidth[7] environment such as Frame Relay, ATM, or others.

Windows NT 5.0 uses site information to locate an Active Directory server close to the user. When a user workstation connects to the network, it receives a TCP/IP address from a DHCP server, which also identifies the subnet to which the workstation is attached. Workstations that have statically configured IP addresses also have statically configured subnet information. In either case, the Windows NT 5.0 domain controller (DC) Locator will attempt to locate an Active Directory server located on the same subnet as the user, based on the subnet information known to the workstation.

The Windows NT 5.0 replication system automatically generates a ring topology for replication among Active Directory servers in a given site. Within a site, directory replication is performed via remote procedure call (RPC). Between sites replication can be selectively configured to use RPC or messaging. Windows NT 5.0 provides simple SMTP messaging as a standard feature. If Microsoft Exchange is available, inter-site replication can be carried via

[7] “High bandwidth” means ETHERNET speed (10 million bits per second) or better in this context

Exchange, using any of the many mail transports supported by Exchange which include SMTP, X.400, and others.

A minimal site consists of a single IP subnet. Windows NT 5.0 assumes that all machines located in the same site share a common high-bandwidth network. Given this, a good site design is one in which all subnets assigned to a given site in fact share such a network. Areas of a network that are separated by a wide area network, multiple routers, or other slower links should be in separate sites.

Schema

The Active Directory schema defines the set of all object classes and attributes that can be stored in the directory. For each object class, the schema defines where it can be created in a directory tree by specifying the legal parents of the class. The content of a class is defined by the list of attributes that the class must or may contain.

Users and applications extend the schema when there are no existing object classes that meet the need at hand. Extending the schema is a simple, straightforward process.

New attributes can be added to the schema at any time. An attribute definition consists of a name, a unique Object Identifier (OID), a syntax that defines what kind of data the attribute can hold, and optional range limits. For strings, the value limits set the minimum and maximum length of the string. For integers, the value limits set the minimum and maximum value of the integer.

Query performance in the Active Directory is directly related to the availability of an index that can be used to optimize a given query. When no index is available to satisfy a given query, the LDAP server must read the entire partition to satisfy the query. When you define an attribute, you have the option of creating an index for that attribute. It is also possible to create an index over a given attribute by setting the **searchFlags** attribute in the **attributeSchema** object to 1. You should define an attribute as indexed when:

- The attribute will frequently be used in queries. Adding an index consumes storage and affects insert performance (because the index must be maintained when an item is inserted), so you should not add an index if it will not be used often.
- The values in the attribute will be highly unique. Boolean attributes should never be indexed, because a boolean attribute will only have two possible values: True or False. Employee numbers and surnames are highly unique and therefore make

good indexes.

- The attribute will occur on objects of interest. Indexing an attribute allows you to quickly find objects that have that attribute instantiated. Before adding an index, make sure the attribute you are indexing is a must-have or may-have on the objects you wish to retrieve.

New object classes can be added to the schema at any time. An object definition consists of a name, an Object Identifier (OID), a list of may-contain and must-contain attributes, the list of classes that can be parents of the object, the class the object is derived from, and a list of any auxiliary classes that apply to the object.

Because the schema controls what can be stored in the directory, and describes what is already stored, write access to the schema is limited to administrators by default. A schema management snap-in for the Microsoft Management Console (MMC) is provided with Windows NT 5.0. To extend the schema, a suitably privileged user can create new attributes and classes. Attributes can then be added to new or existing classes. For each new attribute or class, an OID is required.

An Object Identifier (OID) is a number unambiguously identifying an object class or attribute in a directory service[8]. OIDs are issued by *issuing authorities* and form a hierarchy. An OID is represented as a dotted decimal string (for example, “1.2.3.4”). Enterprises (and individuals) can obtain a root OID from an issuing authority and use it to allocate additional OIDs. For example, Microsoft has been issued the root OID of 1.2.840.113556. Microsoft manages further branches from this root internally. One of these branches is used to allocate OIDs for Active Directory classes, another for Active Directory attributes, and so on.

Many countries in the world have an identified *National Registration Authority (*NRA) responsible for issuing OID’s to enterprises. In the United States, the NRA is the American National Standards Institute (ANSI). The NRA issues root OIDs. An enterprise can register a name for the OID as well. There is a fee associated with both root OIDs and registered names. Contact the NRA for your country for details[9].

Publishing

Publishing is the act of creating objects in the directory that either

[8] OIDs are used in many other applications as well, but OIDs are best known for their use in directory service applications.

[9] The International Standards Organization (ISO) maintains a list of NRAs on their web site at http://www.iso.chorg.

directly contain the information you wish to make available or provide a reference to the information you wish to make available. For example, a user object contains useful information about users, such as their telephone numbers and e-mail addresses, while a volume object contains a reference to a shared file system volume.

Information should be published in the Active Directory when it is useful or interesting to a large part of the user community and when it needs to be highly accessible.

Information published in the Active Directory has two major characteristics:

- It is relatively static and changes infrequently. Telephone numbers and e-mail addresses are examples of relatively static information suitable for publishing; the user's currently selected e-mail message is an example of highly volatile information.
- It is structured and can be represented as a set of discrete attributes. A user's business address is an example of structured information suitable for publishing; an audio clip of the user's voice is an example of unstructured information better suited to the file system.

Operational information used by applications is an excellent candidate for publishing in the Active Directory. This includes global configuration information that applies to all instances of a given application. For example, a relational database product could store the default configuration for database servers as an object in the Active Directory. New installations of that product would collect the default configuration from the object, simplifying the installation process and enhancing consistency of installations in an enterprise.

Applications can also publish their *connection points* in the directory. Connection points are used for client-server rendezvous. The Active Directory defines an architecture for integrated service administration using *Service Administration Point* objects, and provides standard connection points for RPC, Winsock, and COM applications. Applications that do not use the RPC or Winsock interfaces for publishing their connection points can explicitly publish *Service Connection Point* objects in the directory.

Application data can also be published in the directory using application-specific objects. Application-specific data should meet the criteria discussed above, that is, it should be globally interesting, relatively non-volatile, and structured.

RPC—RPC applications use the RpcNs* family of APIs to publish their connection points in the directory and to query for the connection

points of services that have published theirs.

Windows Sockets—Windows Sockets applications use the Registration and Resolution family of APIs available in Winsock 2.0 to publish their connection points and query for the connection points of services that have published theirs.

DCOM—DCOM services publish their connection points via the ~~DCOM~~ Class Store, which resides in the Active Directory.

Groups

Windows NT 5.0 introduces new Group features

- Groups can be treated as distribution lists if the next major release of Exchange is installed
- Groups can contain non-security members (important when the group is used for both security and distribution list purposes)
- Security use of a groups can be disabled (important when the group is solely used as a distribution list).
- Groups can be nested
- A new group type, the Universal group is introduced.

A **Universal group** is the simplest form of group. Universal Groups can appear in ACLs anywhere in the Forest, and can contain other Universal Groups, Global Groups, and Users from anywhere in the forest. Small installations can use Universal groups exclusively and not concern themselves with Global and Local groups.

A **Global group** can appear on ACLs anywhere in the Forest. A global group can contain users and other global groups from its own domain.

A **Domain Local group** can be used on ACLs only it its own domain. A domain Local group can contain users and global groups from any domain in the forest, universal groups, and other Domain Local groups in its own domain.

The three group types provide a rich and flexible access control environment while reducing replication traffic to the Global Catalog caused by group membership changes. A Universal group appears in the GC, but will contain primarily global groups from domains in the forest.. Once the global groups are established, the membership in the Universal group will change infrequently. Global groups appear in the GC, but not their members. Membership changes in global groups are not replicated outside of the domain where they are defined. Domain Local groups are valid only in the domain where they are defined and do not appear in the GC at all.

MIGRATION

This section presents an overview of migration from earlier versions of Windows NT. Migration is discussed in detail in a separate document, “Migrating from Windows NT 3.51 and 4.0 to Windows NT 5.0”.

Supported Upgrade Paths

You can upgrade Windows NT 3.51 and 4.0 systems directly to Windows NT 5.0. Windows NT 3.1 and 3.5 systems must be upgraded to Windows NT 3.51 or 4.0 before they can be upgraded to Windows NT 5.0. A new installation of Windows NT 5.0 can be performed on any system listed in the *Windows NT 5.0 Hardware Compatibility List*.

Domain Controllers

In Windows NT 3.51 and 4.0 there are two kinds of Domain Controllers–Primary Domain Controllers (PDCs) and Backup Domain Controllers (BDCs). Domain Controllers keep a copy of the directory. Primary Domain controllers hold a read/write copy while backup domain controllers hold a read-only copy.

In Windows NT 5.0, all Domain Controllers for a given domain hold a writeable copy of the directory. There is no distinction between primary and backup; all domain controllers are equal.

To migrate an Windows NT 3.51 or 4.0 domain to Windows NT 5.0:

Upgrade the Primary Domain Controller for the domain to Windows NT 5.0. This automatically loads the users and groups from the domain directory into the Active Directory. As part of the upgrade process you specify whether this domain will be:

- The root of a new tree in a new forest
- The root of a new tree of domains in an existing forest
- A child of an existing tree of domains

The domain is now a *mixed domain*. You can use the Windows NT 5.0 administration tools to manage the domain, and you can create a hierarchical structure of *Organizational Unit* folders in the directory to delegate administrative authority. Nested groups are not supported in mixed mode. ~~, but changes must occur at the Primary Domain Controller. This is because the remaining Backup Domain Controllers have not been upgraded to Windows NT 5.0 and do not understand multi-master replication.~~ The Backup Domain Controllers, member servers, and clients are unchanged and unaware that the PDC is now an Active Directory server.

Then migrate Backup Domain Controllers by upgrading them to Windows NT 5.0. The upgrade process will recognize the Backup

Domain Controller, automatically install it as an Active Directory replica, and insert it into the replication topology. When all domain controllers have been upgraded to Windows NT 5.0, the domain is no longer a mixed domain, ~~and changes can be made to any replica~~and nested groups are supported.

Member Servers

Member servers are migrated by upgrading them to Windows NT 5.0. Local users and local groups are stored in the registry of the member server and are not moved into the Active Directory. Upgrading a member server to Windows NT 5.0 allows it to participate in Kerberos security, adds support for Active Directory-aware applications, and adds the Microsoft Management Console and Active Directory-aware shell and common dialogs.

Clients

Windows NT Workstation clients are migrated by upgrading them to Windows NT 5.0. Windows 95 clients are migrated with the installation of a service pack containing the additional software components needed to make them Active Directory-aware. Upgrading a client allows it to participate in Kerberos security, adds support for Active Directory-aware applications and adds the Active Directory-aware shell and common dialogs.

User Accounts

When a downlevel PDC (3.51 or 4.0) is upgraded to Windows NT 5.0, the users are migrated to the Active Directory and placed in a container in the domain root called "Users".

Machine Accounts

When a downlevel PDC (3.51 or 4.0) is upgraded to Windows NT 5.0, the machine accounts are migrated to the Active Directory and placed in a container in the domain root called "Computers".

Global Groups

When a downlevel PDC (3.51 or 4.0) is upgraded to Windows NT 5.0, the Groups are migrated to the Active Directory and placed in a container in the domain root called "Users". Built-in groups are in a special container called "Built-in".

FREQUENTLY ASKED QUESTIONS

How does a workstation discover its site?

A workstation discovers it's site by presenting it's subnet to the first Active Directory server contacted. The workstation determines the subnet by applying its subnet mask to its IP address. The subnet mask and IPO address can be assigned by DHCP or statically configured. The first server contacted uses the presented subnet to locate the Site object for the site where the workstation is located. If the current server is not in that site, the server will notify the workstation of a better server to use.

How does a workstation find a directory server?

A workstation finds a directory server by querying DNS. Directory servers for a given domain publish SRV Resource Records in DNS with names in the form "LDAP.TCP.<domain name>. Thus, a workstation logging in to Microsoft.Com will query DNS for SRV records for LDAP.TCP.Microsoft.Com. A server will be selected from the list and contacted. The contacted server will use the subnet information presented by the workstation to determine the best server as described in the previous question.

How do I log in?

A user can log in with a Windows NT 5.0 workstation with names in a variety of formats. These include the name formats supported by the Win32 API “DsCrackNames”, which are used to convert among these name forms.

- Domain NETBIOS name and SAM-Account-Name—This is the Windows NT 4.0-style login. The domain NETBIOS name is the name the domain had prior to migration. The SAM-Account-Name is the account name the user had prior to migration.
- User Principal Name, i.e.<friendly-name>@<dotted-dns-domain-name>—~~The left-hand component must be unique in the specified scope. The scope is defined first by <dotted-dns-domain-name> and second by whether the DC selected is a global catalog or not. The DSA will refer the search to the named domain if it is not the local domain. If the DSA hosts a GC, it will search the GC.~~ If the name is not unique the login attempt will fail with an Unknown User error.
- ~~RFC 1779 Name—For example, CN=JamesSmith,CN=USers,DC=PBS,DC=Microsoft,DC=Com,O=Internet.~~
- ~~Canonical Name—For example,~~

Part 1: Local Groups on Member Servers

~~pbs.microsoft.com/Users/JamesSmith.~~

- ~~String GUID form of the user object's unique identifier—For example, {4fa050f0-f561-11cf-bdd9-00aa003a77b6}.~~

What happens to Access Control Lists on domain resources after migration?

Access control lists are not affected directly by migration. If all Windows NT 3.51 and Windows NT 4.0 domains are migrated in place, nothing changes from an ACL perspective.

If you move servers from a resource domain into an organizational unit in migrated account domains *and delete the resource domain* you will need to edit any ACLs that hold ACEs referring to the (now deleted) domain. This is not a function of the Windows NT 5.0 migration; if you delete a domain in any version of Windows NT, security identifiers issued by that domain become invalid.

To reduce the effort of reapplying ACL resources, if you have a resource domain in Windows NT 3.51~~2~~ or 4.0 and you plan to replace it with an OU and delete it in Windows NT 5.0, you should not put groups from the resource domain into ACLs. Note that this does not affect local groups from member servers, only from DC's.

Microsoft will provide tools to assist in re-ACLing resources as part of Windows NT 5.0.

What happens to ACL's when I delete a domain?

When you create a group in a domain, it has a Security Identifier (SID) issued by that domain. When you put that SID into an ACL, it grants access to users who have that SID in their token. Users get the SID in their token by logging on to the domain that issued the SID. This can be a network logon and happen transparently.

When you edit an ACL, the LookupAccountName API is called with the SID. If you delete the domain that issued the SID, you will see "Unknown User" in the list of users and groups for the ACL. This happens in Windows NT 3.51 and Windows NT 4.0 if you delete a domain or remove a trust link.

What happens to local groups?

There are two parts to this answer:

Local groups on a member server stay local; they exist only in the SAM on the member server, and are not migrated into the Active Directory. Typical use of a local group on a member server is to hold global groups from account domains. The server administrator puts

the local group in the ACLs on the server's resources, and adds the global groups to the local group. This does not change in Windows NT 5.0. The only difference is that the global groups become normal groups and are published in the Active Directory.

BDC's have a read-only SAM. When you create a local group on a BDC the create operation is remoted to the PDC and replicated back to all BDC's. Semantically these local groups are identical to the local groups on a member server, but they exist on the PDC and all BDC's. When you upgrade to Windows NT 5.0, the upgrade process creates ~~a LocalGroup~~ a Domain Local group object in the Active Directory for each of these~~. LocalGroup objects are only special in the sense that you can't create new ones. New groups are simply Windows NT groups published in the Active Directory.~~

When is the Global Catalog searched?

A search of the global catalog is initiated in one of several ways:

- By a subtree or one-level LDAP search rooted at the null DN (the root of the namespace). This will generate a referral to the Global Catalog.
- By a direct reference to the GC port at a GC replica (although client programs are unlikely to take this approach).
- By an explicit reference to the GC ADSI provider ("GC://").

Do I have to use Microsoft’s DNS Server?

No. There are significant advantages to using Microsoft DNS, but any RFC-compliant DNS server will work. Dynamic DNS is recommended, because Active Directory servers can automatically register the necessary records in DNS given a Dynamic DNS server. Static DNS servers will work equally well, but the DNS registration for each Active Directory server must be accomplished manually.

What are the advantages of using Microsoft’s DNS Server?

The DNS server included with Windows NT 5.0 is an RFC and BIND compliant implementation of Dynamic DNS. It is a native implementation for Windows NT, not a port of the public domain BIND implementation. The Microsoft DNS server stores the DNS zones for which it is authoritative in the Active Directory. DNS data is replicated among Microsoft DNS servers by Active Directory replication, not Zone Transfer. The Microsoft DNS server supports standard DNS Zone Transfer for interoperability with other DNS servers.

FOR MORE INFORMATION

What happens to DHCP?

The DHCP server is largely unchanged for Windows NT 5.0. The DHCP client is DNS-aware and uses the services of Dynamic DNS to register addresses issued by DHCP directly in DNS. The DHCP client will continue to register with WINS, if a WINS server is identified by DHCP.

What happens to WINS?

WINS is unchanged for Windows NT 5.0. Windows NT 5.0 clients (and Windows 95 clients with the Active Directory upgrade installed) no longer need to use the NETBIOS namespace. WINS is still required for downlevel clients to find servers and vice versa. When there are no more downlevel clients in the enterprise, the WINS servers can be turned off.

For the latest information on Windows NT Server, check out our World Wide Web site at http://www.microsoft.com/backoffice or the Windows NT Server Forum on the Microsoft Network (GO WORD: MSNTS).